

一种基于连接树表的关系数据库关键词查询方法

莫红枝, 甘井中, 吕 洁

(玉林师范学院 教育技术中心, 广西 玉林 537000)

摘 要: 基于连接树表提出一种新的方法以提高关系数据库关键查询的效率: 从模式图中找到所有节点小于 Max-size 的连接树以及对应的 select 模板, 将其存储在连接树表 joinTree 中去, 再使用 KSORD 的搜寻方法来寻找出候选连接树的 select 模板, 从而改善现有 KSORD 过程所涉及到的广度优先遍历算法的搜索效率. 最后通过实验验证, 本方法可明显缩短查询时间.

关键词: 关系数据库; 模式图; 连接树; 元组

中图分类号: TP39

文献标识码: A

文章编号: 1000-7180(2014)06-0117-04

A Keywords Search Method over Relational Databases Based on Join Tree

MO Hong-zhi, GAN Jing-zhong, LV Jie

(School of Computer Science, Yulin Normal University, Yulin 537000, Guangxi)

Abstract: A solution based on join tree is proposed to improve the efficiency of keywords search over relational databases (KSORD). In the solution, a join tree and a select template are generated from a schema graph. During a query of KSORD, select template of the candidate join tree can be found out efficiently, instead of finding it from a schema graph by Breadth-First Search. Finally, an experiment is verified that the solution is more efficiently than the current KSORD.

Key words: relational databases; schema graph; join tree; tuple

1 引言

关系数据库关键词查询基于信息检索技术和关系数据库查询技术, 以实现关系数据库关键词的自由态查询. 当前 KSORD 的研究方法往往采用模式图方法, 比较经典的系统有: DBXplorer^[1]、DISCOVER^[2] 和 ITREKS^[3] 等, 采用模式图这个搜寻方法的搜寻过程可以分为三个步骤, 这三个步骤分别为: 生成元组集的阶段、生成候选连接树的阶段以及元组连接树阶段. 然而候选连接树往往采用广度优先遍历算法从模式图中得到, 这就导致候选连接树生成阶段耗时过长, 影响整个查询效率. 针对这个问题, 本文提出了用连接树表的方法来解决这个问题: 使用 PageRank^[4] 模式图的方法在关系数据库中寻

找节点和边赋予权值; 从模式图中找到所有节点数不大于 Maxsize 的连接树以及对应的 select 模板, 将其存储在连接树表 joinTree 中, 那么此时通过 KSORD 方法搜寻, 可以直接地从 joinTree 连接树表中找出候选的连接树 select 模板, 那么这种方法就可以替代了以往的 KSORD 方法, 以往的 KSORD 方法是利用广度优先的遍历方法从模式图去寻找候选连接树, 这样的方法的效率比较低. 最后通过一个数据实例该解决方案在运行时间上比以往的 KSORD 的运行时要短.

2 背景

2.1 关系数据库关键词查询

模式图 (G_s) ^[5]: 对于有向图 $G_s = (V, E)$, 其中

收稿日期: 2013-09-22; 修回日期: 2013-11-08

基金项目: 广西壮族自治区教育厅科研项目(201204LX346, 200911LX348)

顶点集 V 为关系数据库中 n 个关系 $v_1 \cdots v_n$ 的集合, 而若 v_i, v_j 由主外键连接, 则 $\langle v_i, v_j \rangle$ 为弧集 E 的元素.

连接树 $(T)^{[5]}$: 一种树型结构, 它的特点是以关系数据库中的关系 $R_1 \cdots R_n$ 作为树的结点, 且每对相邻的结点 R_i 和 R_j , 在模式图 G_s 对应的无向图 G_u 中都能找到对应的边 (R_i, R_j) .

元组连接树 $(t)^{[6]}$: 一种树型结构, 它的特点是以关系数据库中的元组 $t_1 \cdots t_n$, 且对于两个相邻的结点 t_i 和 t_j , 其对应的关系 R_i 和 R_j (即 $t_i \in R_i, t_j \in R_j$) 也是相邻的结点 (即 $t_i \in t_j \in R_i \in R_j$).

关键词查询^[6]: 对于提供的一组有限关键词集合 $K(k_1 \cdots k_n)$, 查询结果所有可能的元组连接树, 其特点是叶子结点至少含有一个 K 中的元素.

top-k 关键词查询^[6]: 设 $K(k_1 \cdots k_n)$ 为一组关键词, 这组关键词是有限的, 得到的查询结果是有 K 个按其评分降序排列的连接树列表, 这 K 个降序排列的连接树列表属于关键词查询的子集, 其评分的标准是按照评分公式 $\text{Score}(T, K)$ 得到的, 并且这个评分公式具有特殊性.

基于模式图的方法, 一般采取的步骤大致如下: 利用全文索引对每个文本属性进行关键词查询, 并返回所查找到结果的关系名; 根据模式图和 A 步中得到的关系名, 生成对应的连接树; 根据上述得到的关键词组、对应的关系名及连接树, 得到元组连接树以及每个元组连接树的评分结果; 根据元组连接树的评分进行排序, 得到 top-k 关键词查询结果.

2.2 PageRank 算法

PageRank 算法是 Google 搜索引擎网页排名的核心算法. 网页是根据其 PR 值进行排名, PR 值越高, 该网页的排名越向前. 如果数量越多的网页链接该页面, 并且 PR 值越大的网页链接该页面, 那么该页面的 PR 值越大.

PageRank 算法利用了 Web 中的网页所对应的图. 该图由邻接矩阵 W 表示, 将矩阵 W 进行归一化处理得到矩阵 W' , 矩阵 W' 进行转置得到转移概率矩阵 A , 矩阵 A 的最大特征值对应的特征向量 X 就是 Web 中网页的 PR 值组成的向量.

3 解决方案

3.1 关系数据库模式图的处理研究

在关系数据库无向模式图中, 对模式图中的各个节点赋上一个值作为它的权重, 这个赋值过程使

用到 PageRank 算法. 下面以图 1 作为例子来介绍使用 PageRank 算法来为关系数据库中的无向模式图的各个节点和表赋予权重值的过程, 并按照这个权重值的大小来对每一张表进行一个降序排序.

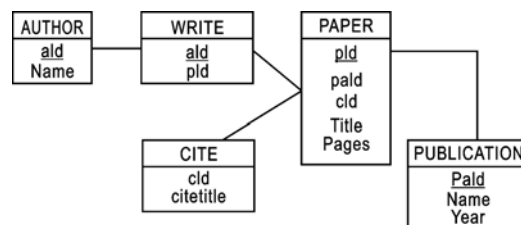


图 1 关系数据库的模式图

图 1 中对应的模式图的邻接矩阵为 W , 通过 PageRank 算法归一化处理得到 W' , 然后转置得到转移概率矩阵 A .

$$W = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$W' = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$A = \begin{pmatrix} 0 & \frac{1}{2} & 0 & 0 & 0 \\ 1 & 0 & \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 1 & 1 \\ 0 & 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & \frac{1}{3} & 0 & 0 \end{pmatrix}$$

求得矩阵 A 的最大特征值 λ 及其特征向量 X : $\lambda = 1$, $X = (1, 2, 3, 1, 1)^T$. 因此, 图 1 模式图中的表按重要程度的排序: PAPER > WRITE > AUTHOR > CITE > PUBLICATION.

对模式图中的边 $E(V_1, V_2)$ 增加权重因子 $\text{weight}(V_1, V_2)$:

$$\text{weight}(V_1, V_2) = \frac{1}{PR(V_1) + PR(V_2)}$$

由上面公式可看出, $\text{weight}(V_1, V_2)$ 为节点 V_1 、节点 V_2 的 PR 值之和倒数, 即两个节点的 PR 值数

值越高它们直接相连的代价越低,在生成代价最小的连接树时该路径越容易被选中.对图1的模式图进行节点和边的处理之后,得到带权重的模式图如图2所示.

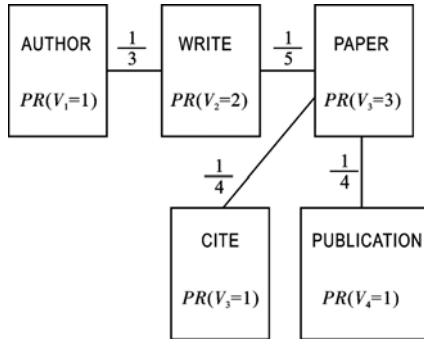


图2 权重模式图

3.2 连接树表的生成研究

为了得到连接树表 joinTree,那么我们认为需要生成连接树对应的 select 模板,而需要生成的连接树则是从使用 PageRank 算法得到的具有权重的模式图中寻找节点数目小于或等于 Maxsize 的子图作为最小生成树.最后是生成 select 模板,这可以分成两个步骤来解决:第一,从带权重的模式图中去寻找其所有节点数目小于或等于 Maxsize 的子图;第二,从这些连通子图中找出最小生成树所对应的 select 模板.

因为模式图通常为稀疏图,所以其子图大多为非连通图.而最小生成树是连通图,有必要根据定理1可将部分非连通图剔除.

定理1 设图 $G(v, e)$ 的节点数为 $N(v)$,边数为 $N(e)$;若 $N(v) < N(e) + 1$,则图 $G(v, e)$ 为非连通图.

可以利用递归思想从模式图中找出节点数不大于 Maxsize 的子图的,算法1给出了一个从模式图中找出节点数不大于 Maxsize 子图的算法.

算法1:从模式图中 $G(v, e)$ 找出结点数不大于 Maxsize 的子图

Input: $G(v, e)$ 中的结点 $V_i, i = 1, 2, 3, \dots, n; \text{Maxsize}$;
 Onput: 节点数不大于 Maxsize 中子图的节点集;
 Begin
 1 ElmentSet (n, k)
 2 for 指针 pos 从集合末向前遍历 k 次
 3 将 pos 指向的数值存入容器 vectSet 中;
 4 if vectSet.size() < k
 5 then ElmentSet ($n-1, k-1$)
 6 elseif vectSet 中的节点组成的子图符合定理1

理1

7 then continue

8 else Ouput vectSet

9 for 指针 j 从1遍历到 Maxsize

10 执行函数 ElmentSet (n, k)

End

在算法1中,ElmentSet (n, k) 是从节点为 n 的模式图中找出结点数为 k 的子图.递归该函数即可得到.节点数不大于 Maxsize 中子图的节点集.

最小生成树算法可解决从子图中生成最小生成树对应的 select 模板问题.算法2利用 Kruskal 给出了一个解决方案.

算法2:根据模式图 $G(V, E)$ 得到最小生成树对应 select 模板

Input: 模式图 $G(V, E)$;

Ouput: Select 查询语句模板

Begin

1 初始化:

$G(V, E)$ 边的权重变量为 key;

Select 模板 strSelect = "where";

2 对边 $E(V_i, V_j)$ 按权重快速排序,并存入 $E[N_e - 1]$;

3 for 用 point 遍历 $E[N_e - 1]$

4 if $E[\text{point}]$ 的两节点存在两个连通分量

5 将 $E[\text{point}]$ 加入到 E' 中;

6 合并连通分量集合;

7 else continue

8 if E' 元素个数等于 $N_v - 1$

9 for 遍历 $E'(V_i, V_j)$

10 strSelect.append(" $V_i, \text{key} = V_j, \text{Key}$ ");

11 return Selector;

12 else return NULL;

End

在算法2中,第7行判断集合 E' 所组成的子图为连通图;第9行是根据节点(表名),键的属性得到 select 模板.而对只有1个节点子图,select 模板为空.对 n 个节点的连接树 $G(V, E)$,其评分如下公式:

$$\text{weight}(G(V, E), n) = \frac{1}{\sum_{i=1 \rightarrow n} \frac{1}{PR(V_i)}}$$

由上述公式可知,连接树的权重值与节点数成反比,并且其值不大于节点集中最小权重值.

算法1和算法2可以从关系数据库模式图中找到连接树对应的 select 模板.将这些连接树的头节点、尾节点、所有节点集、select 模板、权重及其节点数分别存入 joinTree 表的列 IDx、FirstTable、LastTable、AllTable、Selector、weight、size 中,并对头节

点、尾节点建立组合索引,并对 AllTable 创建全文索引.图 1 的模式图通过处理最终得到节点数不大于 5 的下连接树表 joinTree 如图 3 所示.

Seq	FirstTable	LastTable	AllTable	Selector	weight	size
1	AUTHOR	AUTHOR	AUTHOR	NEA	1	1
2	AUTHOR	CITE	AUTHOR,WRITE,PAPER,CITE	where AUTHOR.aId=...	0.353	4
3	WRITE	CITE	WRITE,PAPER,CITE	where WRITE.pId=...	0.535	3
4	PAPER	CITE	PAPER,CITE	where PAPER.cId=...	0.75	2
5	CITE	CITE	CITE	NEA	1	1
6	AUTHOR	PAPER	AUTHOR,WRITE,PAPER	where AUTHOR.aId=...	0.545	3
7	WRITE	PAPER	WRITE,PAPER	where WRITE.pId=...	1.4	2
8	PAPER	PAPER	PAPER	NEA	1	1
9	AUTHOR	PUBLICATION	AUTHOR,WRITE,PAPER,PUBLICATION	where AUTHOR.aId=...	0.353	4
10	WRITE	PUBLICATION	WRITE,PAPER,PUBLICATION	where WRITE.pId=...	0.535	3
11	PAPER	PUBLICATION	PAPER,PUBLICATION	where PAPER.pId=...	0.75	2
12	PUBLICATION	PUBLICATION	PUBLICATION	NEA	1	1
13	WRITE	PUBLICATION	WRITE,PAPER,CITE,PUBLICATION	where WRITE.pId=...	0.353	4
14	PAPER	PUBLICATION	PAPER,CITE,PUBLICATION	where PAPER.cId=...	0.353	3
15	AUTHOR	WRITE	AUTHOR,WRITE	where AUTHOR.aId=...	0.667	2
16	WRITE	WRITE	WRITE	NEA	1	1

图 3 连接树表

查询时可从连接树表中找候选连接树的 select 模板.例如:有一组元组集:AUTHOR {author: Jim}、PAPER {title: DATABASE}、CITE {title: DATABASE};利用 SQL 语句从 joinTree 表直接得到元组集相关属性的查询语句 select 模板:

select Selector,min(weight) where FirstTable="AUTHOR" and LastTable="CITE" and contains(AllTable,"AUTHOR" and "PAPER" and "CITE").

上述 SQL 语句的查询结果为:Selector 为 where AUTHOR. aId=WRITE. aId and WRITE. prId=PAPER. pId and PAPER. cId=CITE. cId, weight 为 0.535;

在元组连接树生成阶段,转化成 SQL 语句为:select * where AUTHOR. aId=WRITE. aId and WRITE. prId=PAPER. pId and AUTHOR="Jim" and WRITE="DATABASE";通过该语句得到 KSORD 的最终结果.

4 实验分析

本实验是在内存为 2 GB 的 Intel Core2 Duo (Merom) T7100(1.8G)的 PC 上进行的,操作系统是 Windows XP,所有代码用 C++实现.为了评测本文提出基于连接树表的 KSORD 解决方案.

本实验首先设计一个模式图,生成连接树表.允许最大候选连接树个数 Maxsize 为 5,8,10,13,15,18 和 20 七个数.每个 Maxsize 值输入 20 个属性组,对比 BFS 方案和 join-Tree 方案生成候选连接树所需要的时间,图 4 表示允许最大候选连接树个数对运行时间的影响.其次设计六个模式图,边数分别为 25,30,35,40,45 和 50.每个模式图输入 10 个属性组,允许最大连接树个数为 6,对比 BFS 方案和 join-Tree 方案生成候选连接树所需要的时间,图 5

表示模式图边数对运行时间的影响.

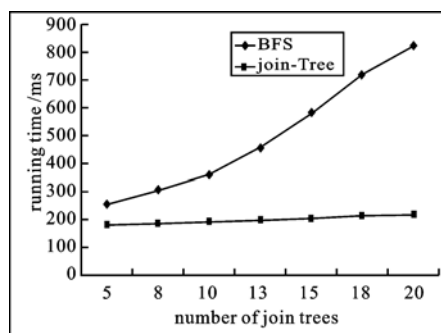


图 4 连接树个数对运行时间的影响

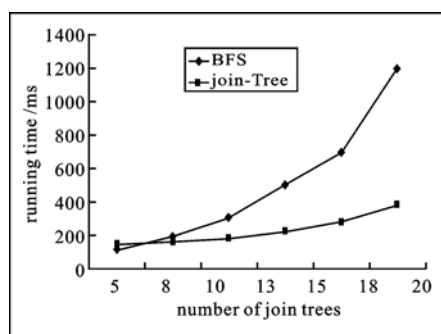


图 5 模式图边数对运行时间的影响

从图 4 可看出,广度优先遍历方案随最大候选连接树个数的增大,其所使用的时间的增长速度快;而相对比我们的方案,随着最大候选连接树个数的增大,其所使用的时间的增长速度比较慢,增加不明显.从图 5 中以看出,随着模式图边的个数增加,对广度优先遍历方案的运行时间增长速度明显大于基于连接树表方案的运行时间,特别是边的个数大到 40 以后,差别更加明显.

5 结束语

本文论述了基于连接树表的 KSORD 查询方法:生成模式图中节点数不大于阈值的子图,再从这些子图中得到最小生成树对应的 select 模板,依据其建立连接树表,KSORD 查询时可高效地利用连接树表得到查询结果.实验结果表明,本方法可明显缩短查询时间.

参考文献:

- [1] Sanjay Agrawal, Surajit Chaudhuri, Gautam Das. DBXplorer: A System for Keyword-Based Search over Relational Databases[C]//Proceedings of the 18th International Conference on Data Engineering. San Jose, CA: IEEE, 2002: 5-16.

(下转第 124 页)

的两步法运行速度提高 4~5 倍左右.

表 2 运用两种算法对图 6(a)和
图 6(b)各检测 20 次的平均时间

	OpenCV 中的算法	本文算法
图 6(a)检测的运行时间	21	5
图 6(b)检测的运行时间	42	8

5 结束语

本文提出用于检测圆形轮廓的改进 Hough 算法. 首先运用标准高斯变换进行去噪处理、Canny 算子进行边缘提取,并在边缘像素点 8 邻域内剔除可能共线的边缘点,以减少后续圆心坐标和圆半径计算时的工作量;然后利用图像中的少数点(边缘点)在一维空间中采用 Hough 变换求取圆心坐标和圆半径. 仿真实验验证了本文所提出的算法的有效性.

参考文献:

- [1] 郑南宁. 计算机视觉与模式识别[M]. 北京: 国防工业出版社,1998.
- [2] Xu L, Oja E. A new curve detection method: Randomized Hough Transform (RHT) [J]. Pattern Recognition Letters, 1990, 11(5): 331-338.
- [3] S H Chiu, J J Liaw, K H Lin, et al. A fast randomized Hough transform for circle/circular arc recognition [J]. International Journal of Pattern Recognition and Artificial Intelligence, 2010, 24(3):457-474.
- [4] Akihiko Torii, Atsushi Imiya. The randomized-hough

- transform-based method for great-circle detection on sphere [J]. Pattern Recognition Letters, 2007, 28 (10): 1186-1192.

- [5] Lianyuan Jiang. Efficient randomized hough transform for circle detection using novel probability sampling and feature points[J]. Optik-International Journal for Light and Electron Optics, 2012, 123 (20): 1834-1840.
- [6] 刘延杰,赖日飞,荣伟彬,等. 基于改进随机 Hough 变换的快速中心检测方法[J]. 纳米技术与精密工程, 2011, 9(4): 298-304.
- [7] 蒋联源,苏勤,祝英俊. 快速随机 Hough 变换多圆检测算法[J]. 计算机工程与应用, 2009, 45 (17): 163-166.
- [8] 张道德,胡新宇,杨光友. 一种改进的随机 Hough 变换圆检测算法[J]. 电子器件, 2009, 32(3): 721-724.
- [9] 秦开怀,王海颖,郑辑涛. 一种基于 Hough 变换的圆和矩形的快速检测方法[J]. 中国图像图形学报, 2010, 15(1):109-115.
- [10] 于永彦,王志坚. 一种从图像中检测多种圆的新方法[J]. 微电子学与计算机, 2011, 28(11): 194-196.
- [11] Christopher Hollitt. A convolution approach to the circle Hough transform for arbitrary radius[J]. Machine Vision and Applications, 2013, 24(4): 683-694.

作者简介:

王红茹 女,(1979-),博士,讲师. 研究方向为智能机器人技术.

丁文 男,(1989-),硕士. 研究方向为机器视觉.

(上接第 120 页)

- [2] Hristidis V, Papakonstantinou Y. DISCOVER: Keyword search in relational databases[C]// Proceedings of the 28th VLD Conference, Hongkong, China; IEEE, 2002:670-681.
- [3] Zhan J, Wang S. ITREKS: Keyword search over relational database by indexingtuple relationship [C] // Proc of the 12th Int'l Conf. on Database Systems for Advanced Applications (DASFAA 2007). Bangkok: Springer-Verlag, 2007: 67-78.
- [4] MARKUS SOBEK. A survey of google's pagerank [DB/OL] [2013-08-22] (2006-05-01). <http://pr.efactory.de>.
- [5] Wen Jijun, Wang Shan. SEEKER: keyword-based information retrieval over relational databases [J]. Journal of Software, 2005, 16(7):1270-1281.
- [6] Jianhua Feng, Guoliang Li, Jianyong Wang. Finding top-k answers in keyword search over relational data-

bases using tuple units[J]. Compute Sci & Technol. 2011, 23(7):1035-1049.

- [7] Bhalotia G, Hulgeri A, Nakhey C, et al. Keyword searching and browsing in database using BANKS[C] // ICDE. San Jose: IEEE, 2002: 431-440
- [8] Agrawal S, Chaudhuri S, Das G. DBXplorer: A system for keyword-based search over relational databases [C]// IDCE. San Jose: IEEE, 2002:5-16.

作者简介:

莫红枝 女,(1981-),本科,实验师,研究方向为计算机网络.

甘井中 男,(1970-),硕士,高级工程师,研究方向为计算机软件.

吕洁 女,(1979-),硕士,副教授,研究方向为计算机研究技术.